0.1 Dimensionality Variables

Number of timesteps	$m : \mathbb{N}$
Number of samples	$n : \mathbb{N}$
State dimension	$p : \mathbb{N}$
Control dimension	$q : \mathbb{N}$

0.2 Specification

Given

Initial state	$x_0 : \mathbb{R}^p$
Initial control action guesses	$T : \mathbb{R}^{m \times q}$, where $T = \begin{bmatrix} \hat{u_1} \\ \hat{u_2} \\ \vdots \\ \hat{u_m} \end{bmatrix}$
Timestep	$\Delta t : \mathbb{R}$
Disturbance covariance matrix	$\Sigma : \mathbb{R}^{q \times q}$ (positive definite)
Set of outcomes	$\psi : \Omega^n = \omega_1, \omega_2, \ldots, \omega_n$
Dynamics	$f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^p$
Cost function	$j : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$
Default control action	$\overline{u} : \mathbb{R}^q$
Temperature	$\lambda : \mathbb{R}$; see note [1]

Let **disturbance distribution** $\varepsilon : \Omega \to \mathbb{R} \to \mathbb{R}^q$ be a brownian process such that $\varepsilon(\cdot)(t) \sim N(0, \Sigma t)$. Then let **disturbance tensor** $\mathcal{P} : \mathbb{R}^{n \times m \times q}$ be the matrix of n sample disturbances, via

$$\mathcal{P} = \begin{bmatrix} \varepsilon(\omega_1)(1 \cdot \Delta t) & \varepsilon(\omega_1)(2 \cdot \Delta t) & \ldots & \varepsilon(\omega_1)(m \cdot \Delta t) \\ \varepsilon(\omega_2)(1 \cdot \Delta t) & \varepsilon(\omega_2)(2 \cdot \Delta t) & \ldots & \varepsilon(\omega_2)(m \cdot \Delta t) \\ \vdots & & \ddots & \vdots \\ \varepsilon(\omega_n)(1 \cdot \Delta t) & \varepsilon(\omega_n)(2 \cdot \Delta t) & \ldots & \varepsilon(\omega_n)(m \cdot \Delta t) \end{bmatrix}$$

and let **sample control trajectories** $\mathcal{T} : \mathbb{R}^{n \times m \times q}$ be

$$\mathcal{T} = \mathcal{P} + \begin{bmatrix} T^T \\ T^T \\ \vdots \\ T^T \end{bmatrix} = \mathcal{P} + \begin{bmatrix} \hat{u_1} & \hat{u_2} & \ldots & \hat{u_m} \\ \hat{u_1} & \hat{u_2} & \ldots & \hat{u_m} \\ \vdots & & \ddots & \vdots \\ \hat{u_1} & \hat{u_2} & \ldots & \hat{u_m} \end{bmatrix}.$$

We can then consider the **continuous state trajectories** $X_i : \mathbb{R} \to \mathbb{R}^p$ such that $X_i(0) = x_0, X_i'(j \cdot \Delta t) = f(X_i(j \cdot \Delta t), u_i)$. This gives us the **discretized state trajectories** $\mathcal{X} : \mathbb{R}^{n \times m \times p}$

$$\mathcal{X} = \begin{bmatrix} X_1(1 \cdot \Delta t) & X_1(2 \cdot \Delta t) & \dots & X_1(m \cdot \Delta t) \\ X_2(1 \cdot \Delta t) & X_2(2 \cdot \Delta t) & \dots & X_2(m \cdot \Delta t) \\ \vdots & & \ddots & \vdots \\ X_n(1 \cdot \Delta t) & X_n(2 \cdot \Delta t) & \dots & X_n(m \cdot \Delta t) \end{bmatrix}.$$

Using the cost function j, discretized state trajectories $\mathcal{X}$, and sample control trajectories $\mathcal{T}$, we compute **cost matrix** $J : \mathbb{R}^{n \times m}$ via

$$J = \begin{bmatrix} j(\mathcal{X}_{1,1}, \mathcal{T}_{1,1}) & j(\mathcal{X}_{1,2}, \mathcal{T}_{1,2}) & \dots & j(\mathcal{X}_{1,m}, \mathcal{T}_{1,m}) \\ j(\mathcal{X}_{2,1}, \mathcal{T}_{2,1}) & j(\mathcal{X}_{2,2}, \mathcal{T}_{2,2}) & \dots & j(\mathcal{X}_{2,m}, \mathcal{T}_{2,m}) \\ \vdots & & \ddots & \vdots \\ j(\mathcal{X}_{n,1}, \mathcal{T}_{n,1}) & j(\mathcal{X}_{n,2}, \mathcal{T}_{n,2}) & \dots & j(\mathcal{X}_{n,m}, \mathcal{T}_{n,m}) \end{bmatrix}.$$

Right multiplying by a lower triangular matrix gives us the **cost to go** matrix $J' : \mathbb{R}^{n \times m}$, with

$$J' = J \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & 1 & \dots & 1 \end{bmatrix}.$$

Similarly we need the **disturbance log probability densities** $D : \mathbb{R}^{n \times m}$ at each step in each trajectory:

$$D = \begin{bmatrix} \ln f_{\varepsilon(\cdot)(1 \cdot \Delta t)}(\mathcal{P}_{1,1}) & \ln f_{\varepsilon(\cdot)(2 \cdot \Delta t)}(\mathcal{P}_{1,2}) & \dots & \ln f_{\varepsilon(\cdot)(m \cdot \Delta t)}(\mathcal{P}_{1,m}) \\ \ln f_{\varepsilon(\cdot)(1 \cdot \Delta t)}(\mathcal{P}_{2,1}) & \ln f_{\varepsilon(\cdot)(2 \cdot \Delta t)}(\mathcal{P}_{2,2}) & \dots & \ln f_{\varepsilon(\cdot)(m \cdot \Delta t)}(\mathcal{P}_{2,m}) \\ \vdots & & \ddots & \vdots \\ \ln f_{\varepsilon(\cdot)(1 \cdot \Delta t)}(\mathcal{P}_{n,1}) & \ln f_{\varepsilon(\cdot)(2 \cdot \Delta t)}(\mathcal{P}_{n,2}) & \dots & \ln f_{\varepsilon(\cdot)(m \cdot \Delta t)}(\mathcal{P}_{n,m}) \end{bmatrix}$$

where $\ln f_{\varepsilon(\cdot)(t)}(\Delta u)$ is the log probability density of a disturbance at time t, via

$$\ln f_{\varepsilon(\cdot)(t)}(\Delta u) = -\ln\left(\sqrt{(2\pi)^q \det \Sigma}\right) - \frac{1}{2} \Delta u^T \Sigma^{-1} \Delta u \quad \text{(see note [2])}$$

from which we can then consider the **trajectory-to-go log probability densities** $D' : \mathbb{R}^{n \times m}$, with

$$D' = D \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ \vdots & & \ddots & \\ 1 & 1 & \dots & 1 \end{bmatrix}.$$

We then specify the jth **control action** $u_j : \mathbb{R}^q$, for $j : \mathbb{N}, 1 \leq j \leq m$,

$$u_j = \frac{\sum_{i=1}^{n} e^{-\frac{1}{\lambda}J'_{i,j} - D'_{i,j}} \cdot T_{i,j}}{\sum_{i=1}^{n} e^{-\frac{1}{\lambda}J'_{i,j} - D'_{i,j}}},$$

and **control tensor** $T' : \mathbb{R}^{m \times q}$ such that $\forall j : \mathbb{N}, 1 \leq j \leq m$, $T_j = u_j$. Then the algorithm

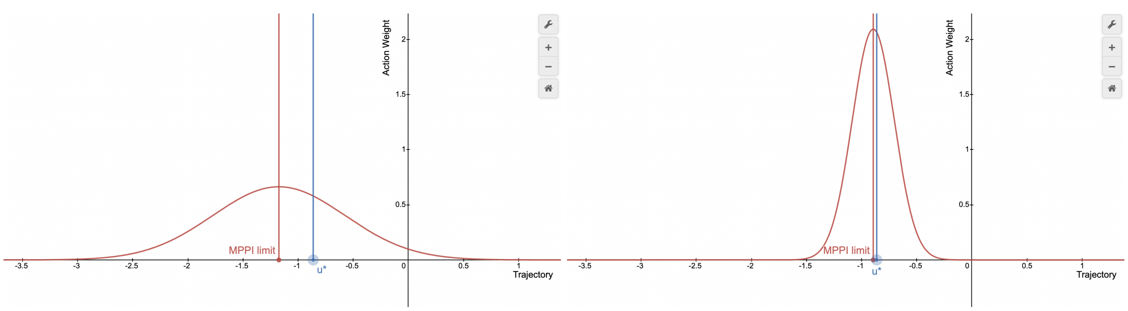
1. Computes T' as above
2. $T \leftarrow \begin{bmatrix} T'_{2:,*} \\ \overline{u} \end{bmatrix}$
3. Returns T'_1

0.3 Notes

1. λ trades optimality for speed and stability. That is, at a high λ MPPI will need fewer samples to converge and/or will be less noisy, but the action MPPI converges to will get no closer to the optimal action as λ increases ($|\mathbb{E}[\text{MPPI}] - u_1^*|$ monotonically increases with λ) *with respect to the dynamics model.* If the model is inaccurate, then increasing the temperature may actaully improve the optimality of the controller.

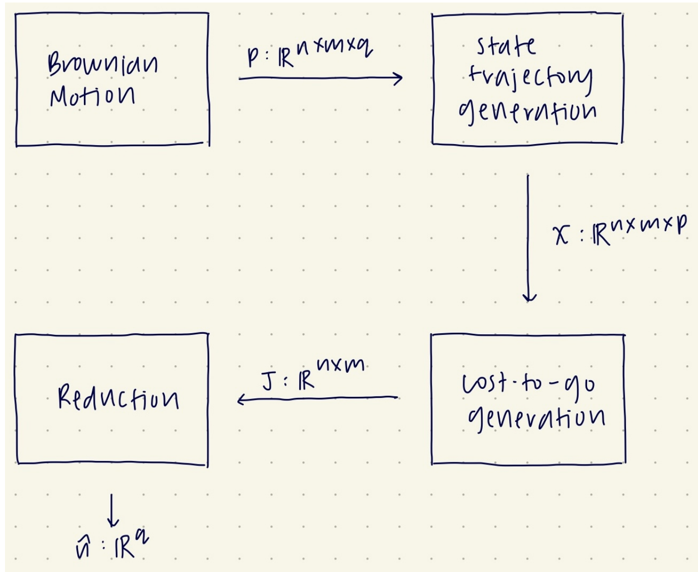
   Tuning plan: decrease until controller is unstable, then increase slowly until performance degrades.

   These two images show the weighting factor ($e^{-\frac{1}{\lambda}J'_{i,j} - D'_{i,j}}$) as a function of the sampled trajectory-to-go at decreasing lambda values:

2. Notice that $\ln\left(\sqrt{(2\pi)^q \det \Sigma}\right)$ is a constant (wrt a single controller step). Since many of the operations here are expensive, pre-computing this is a good idea. If Σ is not scheduled, it is even a compile-time constant!

0.4 CUDA MPPI



We currently have:

- Initial state x_0
- Brownian perturbations $\mathcal{P}$
- Functor (model) f
- Initial action u_0

Process:

1. Matrix that is action dims $\times$ timesteps $(q \times m)$, and add this to each row of the perturbations tensor
    - Do the simulation in a for loop, parallelize across samples

Algorithm 1 Integrate State and Calculate Cost

for i in samples **do**
    $x_{\text{curr}} \leftarrow x_0$
    for j in timesteps **do**
        $u_{ij} \leftarrow \hat{u}_j + \Delta u_i$
        $x_{ij} \leftarrow f(x_{i(j-1)}, u_{ij})$
        $j_{\text{curr}} \leftarrow j_{\text{curr}} - j(x_{ij})$
        $J[j] \leftarrow j_{\text{curr}}$
    control tensor $T'_i \leftarrow w(J_i)$
return T'

- This is equal to the cost to go - the total cost to go
- Need to take T' and sum over samples to get the control tensor T

Reduction: `reduce: (('a * 'b) -> 'b) -> 'b -> 'a list -> 'b`

- Ex. we have arguments `fn(x, y) = x + y, 'b, 1 3 5 8`, then `reduce` returns 17
- With an associative operation, `fn(x,y)` doesn't need to be applied in order (number of operations stays constant, but it can be parallelized)
- The span of the algorithm is $O(\log n)$